TEMA 1: EFICIENCIA DE LOS ALGORITMOS

0. Notación
1. Medida de la complejidad
2. Análisis por casos
3. Notación asintótica

3. Notación asintótica

- **El factor que más interesa de las funciones de coste es la *tasa de crecimiento* para valores grandes de la talla del problema.**
- **Supongamos dos algoritmos A y B, que resuelven el mismo problema con costes:**
  - $T_A(n) = 10n + 4$
  - $T_B(n) = 2n^2 + 2$.

  ¿Cuál es mejor?
- **Depende de la talla del problema, para n<6 es mejor el B, pero para n>=6 es mejor el A.**

  ¿Cuál elegirías?

3. Notación asintótica

- **Comparación de los tiempos asociados a diferentes funciones coste para un ordenador con una potencia de 1 MIPS:**

COSTE		TALLA	
	10	20	100
lg n	0.004 ms	0.005 ms	0.007 ms
n	0.01 ms	0.02 ms	0.1 ms
n lg n	0.033 ms	0.086 ms	0.66 ms
n^2	0.1 ms	0.4 ms	10 ms
n^4	10 ms	160 ms	1 min 40 s
2^n	1 ms	1.05 s	2.6 EU
n!	3.6 s	76000 años	2 E 128 EU
n^n	2 h 48 m	220 EU	2 E 170 EU

3. Notación asintótica

- **Para comparar algoritmos nos interesa un formalismo que permita comparar tasas de crecimiento ➡ Análisis Asintótico**
  - Compara los costes con la talla tendiendo a infinito.
  - Únicamente sirve para una primera comparación entre algoritmos.

3. Notación asintótica

DEF.(Notación de 'el orden de'): Sea una $f:N \rightarrow R^{\geq 0}$ función arbitraria de los números naturales en los reales no negativos. El conjunto de las funciones **del orden de f(n)**, denominado O(f(n)) – O grande - , se define como:

$$\mathbf{O(f(n)) = \{ g: N \rightarrow R^{\geq 0} \mid \exists\, c \in R^{>0}, n_0 \in N.\ \forall n \geq n_0\ g(n) \leq c\ f(n) \}}$$

Una función g(n) es del orden de f(n) sii $g(n) \in O(f(n))$ y se dice que f(n) es cota superior (asintótica) de g(n)

3. Notación asintótica

DEF.(Notación de 'al menos del orden de'): Sea una $f:N \rightarrow R^{\geq 0}$ función arbitraria de los números naturales en los reales no negativos. El conjunto de las funciones **de al menos del orden de f(n)**, denominado $\Omega(f(n))$ – Omega grande - , se define como:

$$\mathbf{\Omega(f(n)) = \{ g: N \rightarrow R^{\geq 0} \mid \exists\, c \in R^{>0}, n_0 \in N.\ \forall n \geq n_0\ g(n) \geq c\ f(n) \}}$$

Una función g(n) es **al menos del orden de** f(n) sii $g(n) \in \Omega(f(n))$ y se dice que f(n) es **cota inferior** (asintótica) de g(n)

3. Notación asintótica

DEF.(Notación de 'el orden exacto de'): El conjunto de las funciones del orden exacto de f(n), denominado $\Theta(f(n))$ - Theta de f(n) -, se define como el conjunto de las funciones que tienen como cota superior e inferior la misma función f(n) :

$$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$$

3. Notación asintótica

- **Ejemplo:** Demostrar que se cumplen las siguientes relaciones:

  1) $4n \in O(3n^2)$

  2) $3n^2 \notin O(4n)$

  3) $n \lg n \in \Omega(\lg n)$

3. Notación asintótica

DEF.(o pequeña): Se define el conjunto de funciones o pequeña de f(n) – o(n) – al constituido por las funciones que cumplen:

$$g(n) \in o(f(n)) \Leftrightarrow \lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$$

De forma similar se puede poner:

$o(f(n)) = \{ g: N \to R^{\geq 0} \mid \forall\, c \in R^{>0}, \exists\, n_0 \in N.\ \forall n \geq n_0\ g(n) \leq cf(n) \}$

3. Notación asintótica

DEF.(ω pequeña): Se define el conjunto de funciones **ω pequeña** de f(n) – ω(n) – al constituido por las funciones que cumplen:

$$g(n) \in \omega(f(n)) \Leftrightarrow \lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$$

De forma similar se puede poner:

$\omega(f(n)) = \{ g: N \to R^{\geq 0} \mid \forall\, c \in R^{>0}, \exists\, n_0 \in N.\ \forall n \geq n_0\ g(n) \geq cf(n) \}$

3. Notación asintótica

El **orden exacto de f(n) - $\Theta(f(n))$ -** también se puede definir utilizando límites:

$$g(n) \in \Theta(f(n)) \Leftrightarrow \lim_{n\to\infty} \frac{g(n)}{f(n)} = k \neq 0$$

3. Notación asintótica

- La notación o pequeña y ω pequeña representan el mismo conjunto que O y Ω respectivamente eliminando el orden exacto. La forma más sencilla de saber si una función pertenece o no a uno de estos conjuntos será utilizando límites:

$\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0 \Rightarrow$ **$g(n) \in o(f(n))$**

$\lim_{n\to\infty} \frac{g(n)}{f(n)} = k \neq 0 \Rightarrow$ **$g(n) \in \Theta(f(n))$**

$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty \Rightarrow$ **$g(n) \in \omega(f(n))$**

$\Rightarrow$ **$g(n) \in O(f(n))$**

$\Rightarrow$ **$g(n) \in \Omega(f(n))$**

3. Notación asintótica

- **Ejemplo:** Demostrar que se cumplen las siguientes relaciones:

1) $4n \in O(3n^2)$

2) $3n^2 \notin O(4n)$

3) $n \lg n \in \Omega(\lg n)$

4) $10n^2 + 4n - 6 \in \Theta(n^2)$

3. Notación asintótica. Propiedades

- **Relaciones de orden:**
  - $f \in O(g(n)) \leftrightarrow f \leq g$ (se dice que f es asintóticamente menor o igual que g)
  - $f \in o(g(n)) \leftrightarrow f < g$
  - $f \in \Theta(g(n)) \leftrightarrow f = g$
  - $f \in \Omega(g(n)) \leftrightarrow f \geq g$
  - $f \in \omega(g(n)) \leftrightarrow f > g$

3. Notación asintótica. Propiedades

- **Relaciones entre cotas:**
  - $f(n) \in O(f(n))$ (Propiedad reflexiva) (También para Ω)
  - $f(n) \in O(g(n)) \wedge g(n) \in O(h(n)) \Rightarrow f(n) \in O(h(n))$ (Propiedad transitiva) (Para todas las notaciones)
  - $f(n) \in O(g(n)) \Leftrightarrow g(n) \in \Omega(f(n))$
  - $\lim_{n\to\infty} f(n)/g(n) = 0 \Rightarrow O(f(n)) \subset O(g(n))$
  - $O(f(n)) = O(g(n)) \Leftrightarrow f(n) \in O(g(n)) \wedge g(n) \in O(f(n))$ (También Ω)
  - $O(f(n)) \subset O(g(n)) \Leftrightarrow f(n) \in O(g(n)) \wedge g(n) \notin O(f(n))$ (También Ω)

3. Notación asintótica. Propiedades

- **Relaciones de inclusión:**

$\forall$ x, y, a, $\varepsilon \in R^{>0}$

  - $\log_a n \in O(\log_b n)$
  - $O(\log_a^x n) \subset O(\log_a^{x+\varepsilon} n)$
  - $O(\log_a^x n) \subset O(n)$
  - $O(n^x) \subset O(n^{x+\varepsilon})$
  - $O(n^x \log_a^y n) \subset O(n^{x+\varepsilon})$
  - $O(n^x) \subset O(2^n)$

3. Notación asintótica. Propiedades

- **Propiedades de clausura:**

  Los conjuntos de funciones (O, Ω, Θ) de f(n) son cerrados respecto de la suma y la multiplicación de funciones:

  - $f_1(n) \in O(g_1(n)) \wedge f_2(n) \in O(g_2(n)) \Rightarrow f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$
  - $f_1(n) \in O(g_1(n)) \wedge f_2(n) \in O(g_2(n)) \Rightarrow f_1(n) \cdot f_2(n) \in O(g_1(n) \cdot g_2(n))$
  - $f_1(n) \in O(g_1(n)) \wedge f_2(n) \in O(g_2(n)) \Rightarrow f_1(n) + f_2(n) \in O(\max(g_1(n), g_2(n)))$

como consecuencia es que los polinomios de grado *k* en *n* son exactamente del orden de n^k :

$$a_k n^k + ... + a_1 n + a_0 \in \Theta(n^k)$$

3. Notación asintótica. Propiedades

- **Jerarquía de complejidades:**

  Podemos establecer una cierta ordenación de las diferentes series de funciones.

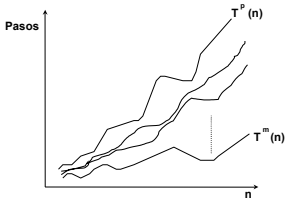
$O(1) \subset O(\lg \lg n) \subset O(\lg n) \subset O(\lg^2 n) \subset O(\sqrt{n}) \subset O(n) \subset O(n \lg n) \subset O(n^2) \subset O(n^k) \subset O(2^n) \subset O(n!) \subset O(n^n)$

constantes poligonales logaritmicas subliniales lineales polinómicas exponenciales

3. Notación asintótica

- **Cálculo del coste asintótico de algoritmos**

  Consideremos un algoritmo con m > 1 instancias, cuyo coste mejor y peor vienen dados por las funciones $T^m(n)$ y $T^p(n)$ respectivamente. Si representamos gráficamente:



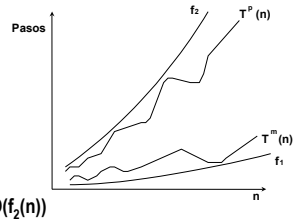
3. Notación asintótica

- **Cálculo del coste asintótico de algoritmos**

  podemos encontrar unas funciones f_1 y f_2 que acoten inferior y superiormente a todas las funciones coste T(n) (una por instancia), de modo que:

  $T^p(n) \in O(f_2(n))$

  $T^m(n) \in \Omega(f_1(n))$



  $T(n) \in \Omega(f_1(n))$ y $T(n) \in O(f_2(n))$

3. Notación asintótica

- **Cálculo del coste asintótico de algoritmos**
  - Sumar_vector: T(n) = 2n + 1
  - Algoritmo B1: $T(n) = n^2 + 3n + 2$
  - Busqueda_lineal:
    - $T^m(n) = 2$,
    - $T^p(n) = 2n + 2$
  - Minimo:
    - $T^m(n) = 2n$,
    - $T^p(n) = 3n - 1$

3. Notación asintótica. Instrucción crítica

- **Utilización de instrucción crítica**
  - Método sencillo de calcular el **coste asintótico en el caso peor**.

DEF. (Instrucción crítica): Es la instrucción que más veces se ejecuta independientemente de los casos.

No siempre existirá una instrucción crítica, pero cuando exista podemos considerar el coste en el peor caso como el número de veces que se repite esta instrucción.

3. Notación asintótica. Instrucción crítica

- **Ejemplo 1**

```
ALGORITMO minimo // Busca la posición del minimo de un vector
DATOS:              a:vector[1..n] de N
RESULTADO:          m:1..n
AUXILIAR:           i:1..n
METODO:
1. m ← 1                                    // 1
2. para i ← 2 hasta n hacer                 // n-1+1
3.   si a[i] < a[m] entonces                 // n-1
4.          m ← i                           // De 0 a n-1 pasos
5.   fsi
6. fpara
fminimo
```

La instrucción crítica será la instrucción *para*, con lo que el coste del algoritmo para el caso peor será: **$T^p(n) = n \in O(n)$**

3. Notación asintótica. Instrucción crítica

- **Ejemplo 2**

```
ALGORITMO A
DATOS:              n: N
RESULTADO:          j: N
METODO:
   j ← n
   mientras j ≠ 0 hacer
     j ← ⌊j/2⌋
   fmientras
fA
```

3. Notación asintótica

- **Ejemplo 3: Analizar el algoritmo de búsqueda dicotómica o binaria en vectores ordenados.**

```
ALGORITMO BUSQUEDA_BINARIA
DATOS:        A: vector[1..n] de N; x : N
RESULTADO:    m: 1..N
METODO:
   izq ← 1
   dch ← n
   Repetir
     m ← [(izq+dch)/2]                          // Div. entera
     si x > A[m] entonces
              izq ← m+1
     sino
              dch ← m-1;
     fsi
   hasta que A[m] = x ∨ izq > dch               // I.C.
   si A[m] ≠ x entonces m ← 0; fsi
fBusqueda binaria
```

3. Notación asintótica

- **Ejemplo 4: Escribir un algoritmo los más eficiente posible que diga si una cadena de bits (vector[1..n] de {0,1}) es o no capicúa. Analizar el algoritmo obtenido en los casos mejor, peor y medio.**

```
ALGORITMO CAPICUA
DATOS:                N: vector[1..n] de {0,1}
RESULTADO:            es: Booleano
METODO:
   (a,b) ← (1,n)
   mientras (a < b) ∧ (N[a] = N[b]) hacer
     (a,b) ← (a+1,b-1)
   fmientras
   si (a >= b) entonces es ← cierto
   sino es ← falso fsi
fCAPICUA
```